уменьшается до 54 бит).

3. Для криптоанализа алгоритма A3A8 можно применить так называемую атаку с “выбранным вызовом” при условии обеспечения физического доступа к SIM-карте. Данная атака требует формирования $1{,}5 \cdot 10^5$ запросов на SIM-карту и занимает около восьми часов.

4. На сегодняшний день лучшей атакой по восстановлению неизвестных ключевых алгоритма A5 является атака “балансировка время-память”. Трудоёмкость данной атаки является величиной порядка 2^{40}.

Литература: 1. M. Briceno, I. Goldberg, D. Wagner, A pedagogical implementation of A5/1, Springer-Verlag, May 1999. 2. A. Biryukov, A. Shamir, Real Time Cryptanalysis of the Alleged A5/1 on a PC, Computer Science department The Weizmann Institute Rehovot, December 1999. 3. J. Golic. Cryptanalysis of Alleged A5 Stream Cipher. - Proceedings of EUROCRYPT'97, LNCS 1233, pp. 239 - 255, Springer-Verlag 1997.

УДК 638.322

ОПТИМІЗАЦІЯ ПРОГРАМНИХ РЕАЛІЗАЦІЙ АЛГОРИТМУ ГОСТ 28147-89

Сергій Коваль, Олександр Тесленко
Національний технічний університет України "КПІ"

Анотація: **На базі визначення системного статусу програм криптографічних перетворень досліджуються методи досягнення максимальної швидкодії програм, які реалізують на сучасних ПЕОМ алгоритми ГОСТ 28147-89.**

Summary: **Based on examination of system status of cryptographic transformation programs, proposed different methods to achieve maximum productivity of GOST 28147-89 based algorithms.**

Ключові слова: **Криптографія, програмування, суперскалярна архітектура.**

I Вступ

В плануванні та проведенні політики інформаційної безпеки одним із найбільш доступних і поширених напрямків є використання криптографічних перетворень. Розвиток інформаційних технологій обумовлює тенденцію включення криптографічних програм у склад системного програмного забезпечення сучасних ЕОМ. На сьогоднішній день відома велика кількість криптографічних стандартів, які використовуються в криптографічних системах. Згідно з вимогами нормативно правових актів із криптографічного захисту інформації, в нашій державі використовується стандарт криптографічного перетворення ГОСТ 28147-89. Потенційний системний статус програм для криптографічних перетворень потребує ретельного й оптимального програмування алгоритмів стандарту із врахуванням особливостей архітектури конкретних ЕОМ. Зростання пропускної здатності фізичних каналів передачі даних у комп'ютерних мережах вимагає адекватного зростання швидкодії криптографічних програм. У зв'язку з цим велике практичне значення мають дослідження методів програмування, які б забезпечували оптимізацію програм за критерієм швидкодії (продуктивності) при реалізації алгоритмів ГОСТ 28147-89 на сучасних ПЕОМ.

II Основна частина

Оптимізація програм за швидкодією потребує від алгоритму виділення тих його частин, які найчастіше виконуються, а від процесора – виділення ефективної підмножини команд і умов їх виконання, які забезпечують мінімальну кількість тактів процесора.

Алгоритми за ГОСТ 28147-89 характеризуються вкладеною циклічністю, де зовнішній цикл забезпечує послідовну обробку 8-байтних блоків вхідних даних і містить один із трьох базових циклів. В тілі будь-якого із базових циклів багаторазово використовується основний крок криптографічного перетворення, який у свою чергу містить цикл

підстановки. Таким чином пріоритет у досягненні швидкодії повинен бути в послідовності - цикл підстановки - основний крок криптографічного перетворення - базовий цикл.

Найбільш поширені в існуючому парку ПЕОМ процесори I486 та Pentium (P5, P6) характеризуються наступними особливостями [1, 2], що істотно впливають на кількість тактів, необхідних для виконання однієї і тієї ж команди:

a) Наявність елементів суперскалярної архітектури, що допускають принципову можливість одночасного виконання команд. На жаль, ці можливості в ряді випадків суттєво відрізняються для різних типів процесорів і не завжди в кращий бік для більш пізніх моделей, що ускладнює універсальність оптимізації.

b) Наявність буферів попередньої вибірки команд та багатоступеневих конвеєрів попередньої обробки команд. Очищення буферів попередньої вибірки командами передачі управління суттєво впливає на продуктивність процесора і потребує додаткових тактів для першої команди при пустому буфері.

c) Наявність вбудованої кеш-пам'яті великої ємності окремо для команд і для даних.

d) Чутливість кількості тактів, необхідних для виконання команди, від місця розташування даних і самої команди в пам'яті. Відсутність вирівнювання адресів кодів команд, на які передається управління, штрафується мінімум 2 тактами для процесора I486. Відсутність вирівнювання адресу даних для процесорів I486 та Pentium штрафується мінімум 2 тактами.

Розглянемо методику створення оптимальних криптографічних програм, які б ґрунтувалися на приведених особливостях алгоритмів і процесорів.

Одним із методів, який враховує особливості процесорів по п. b), полягає у створенні програм без розгалужень ("неветвящихся программ", лінійних програм), досліджених в [3]. Окрім підтримки заповнення конвеєрів і буферів попередньої вибірки такий метод має і додаткові переваги, до яких належать:

- спрощення обчислення ефективної адреси пам'яті для ключів шифрування та блоків підстановки, оскільки частина обчислень може бути виконана при компіляції;
- знищення необхідності у використанні команди LOOP, застосування якої у випадку оптимізації за швидкодією фактично не рекомендується [1] та звільняє регістр ECX;
- знищення необхідності вирівнювання кодів команд.

До недоліків лінійних програм належить потенційне зменшення ефекту використання кеш-пам'яті команд, оскільки кожна команда лінійної програми є "одноразового використання". Але лінійна реалізація навіть базових циклів ГОСТ 28147-89 оцінюється приблизно у 3 Кбайт, що значно менше розміру кеш-пам'яті команд. Таким чином потенційна ефективність використання кеш-пам'яті команд досягається в даному випадку на макрорівні - в процесі криптографічних перетворень даних у цілому, а не одного окремого блоку, тобто в процесі багатократного виконання базових циклів.

Наступний метод підвищення ефективності криптографічних програм пов'язаний з ретельним програмуванням внутрішнього циклу – циклу підстановок і, насамперед, у зменшенні числа ітерацій цього циклу. Стандарт ГОСТ 28147-89 приписує виконання восьми ітерацій циклу підстановок, в кожній з яких реалізується повна чотирибітна підстановка. Для задавання таблиць восьми чотирибітних підстановок достатньо 64 байтів, які є довгостроковим ключем. Очевидною є можливість суміщення ітерацій циклу підстановок за рахунок збільшення розміру таблиць підстановок. Так, у випадку чотирьох таблиць восьмибітних підстановок необхідно виділити 1 Кбайт, а у випадку двох таблиць шістнадцятибітних підстановок – 128 Кбайт пам'яті. Такі розширені таблиці легко формуються із довгострокового ключа, а їх розмір не є занадто великим для систем управління доступом. З точки зору оптимізації необхідно забезпечити розміщення таблиць

ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.

в пам'яті на межі подвійних слів (п. d). Використання розширених таблиць дозволяє аналізувати можливість модифікації алгоритмів ГОСТ 28147-89 шляхом використання незалежних восьмибітних і шістнадцятибітних підстановок.

Перспективним методом оптимізації програм є використання можливостей суперскалярної архітектури процесорів (п. а). Алгоритми по ГОСТ 28147-89 розроблені з орієнтацію на класичну архітектуру, що потребує додаткових зусиль оптимізації програм на суперскалярних процесорах. При послідовній обробці блоків вхідних даних такі зусилля найбільш перспективні для програмування циклу підстановок, ітерації якого за своїм визначенням є паралельними. Це дозволяє шляхом розподілу регістрів між двома сусідніми ітераціями добиватись одночасного виконання команд формування двох різних підстановок. Для зменшення затримки при виході з ітерацій підстановок та забезпечення подальшого паралелізму доцільно поєднати завершення поточного кроку криптографічного перетворення з початком наступного.

З урахуванням вищевикладеного основний крок криптографічного перетворення та цикл восьмибітної підстановки доцільно запрограмувати у вигляді наступної макрокоманди, в якій зірочкою відмічені команди, що можуть виконуватись одночасно з наступною:

```
Step_Substit Macro prim, @m, last
; prim =1 - перший крок базового циклу
;last = 1 - останній крок базового циклу
;@m - номер ключа
;ESI - N1, EDI - N2
;ECX та EBX базові регістри сусідніх ітерацій підстановок,
;       в ці регістри на верхньому рівні програми
;       необхідно записати нульові значення
;EDX та EAX - робочі регістри сусідніх ітерацій підстановок
                If      prim
                Mov     EAX,Key[@m shl 2]
                Add     EAX,ESI
                Endif
;
                Mov     CL, AL                          ;*
                Mov     BL, AH                          ;*
                Mov     DL, LongKey[ECX]        ;*
                Rol     EAX, 16                         ;*
                Mov     DH, LongKey[EBX][256]           ;*
                Mov     CL, AL                          ;*
                Rol     EDX, 16                         ;*
                Mov     BL, AH                          ;*
                Mov     DL, LongKey[ECX][512]           ;*
                Mov     DH,LongKey[EBX][512+256]        ;*, якщо наступна
;                                                       команда Mov
                If      (not prim) and (not last)
                Mov     EAX,Key[@m shl 2]       ;*
                Endif
;
                Ror     EDX, 5                          ;
                Xor     EDX,EDI                 ;* залежно від if
;
                If      not last
                Mov     EDI, ESI                ;*
```

```
                Add     EAX, EDX                ;*
                Endif
;
                Mov     ESI, EDX                ;*
                Endm
```

На базі макрокоманди Step_Substit формуються макрокоманди базових циклів. Наприклад, макрокоманда базового циклу шифрування в режимі простої заміни при умові, що ESI - N1, EDI - N2, буде мати наступний вигляд:

```
Enc32           macro
                Step_Substit    1,0,0
@j1v            =               1               ; Ініціалізація лічильника.
                Rept            7
                Step_Substit    0, @j1v, 0
                Endm
                Rept            2               ; Цикл 2 рази
@j1v            =               0               ; Ініціалізація лічильника.
                Rept            8               ; Цикл 8 разів
                Step_Substit    0, @j1v, 0
@j1v            =               @j1v + 1                ; Наступна ітерація циклу.
                Endm
                Endm
@j1v            =               7                       ; аналогічно
                rept            7
                Step_Substit    0, @j1v, 0
@j1v            =               @j1v - 1
                Endm
                Step_Substit    0, 0, 1
                Endm
```

У випадку шістнадцятибітних підстановок макрокоманда Step_Substit буде мати наступний вигляд

```
Step_Substit Macro prim, @m, last
; prim =1 - перший крок базового циклу
;last = 1 - останній крок базового циклу
;@m - номер ключа
;ESI - N1, EDI - N2
                If      prim
                Mov     EAX,Key[@m shl 2]
                Add     EAX,ESI
                Endif
;
                Mov     BX, AX                          ;*
                Rol     EAX, 16                         ;*
                Mov     DX, LongKey[EBX]        ;*
                Mov     BX, AX                          ;*
                Rol     EDX, 16                         ;*
```


ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.

```
                Mov     DX, LongKey[EBX][65536] ;*, якщо наступна
;                                                 команда Mov
                If      (not prim) and (not last)
                Mov     EAX,Key[@m shl 2]       ;*
                Endif
;
                Ror     EDX, 5                          ;
                Xor     EDX,EDI                 ;* залежно від if
;
                If      not last
                Mov     EDI, ESI                ;*
                Add     EAX, EDX                ;*
                Endif
;
                Mov     ESI, EDX                ;*
                Endm
```

Наступним методом, який забезпечує використання переваг суперскалярної архітектури процесорів Pentium, є організація одночасної обробки двох блоків вхідних даних. На жаль, одночасну обробку двох блоків ГОСТ 28147-89 допускає тільки в двох режимах - режимі простої заміни та режимі гами (без зворотного зв'язку). Для цього необхідно розділить регістри загального призначення між двома блоками. Враховуючи недостатню кількість регістрів, накопичувач N 1 або N 2 для різних блоків необхідно розмістити у пам'яті. З великою ймовірністю вони будуть у кеш-пам'яті, тому така заміна суттєво не впливає на швидкодію. Нехай для першого блоку вхідних даних як і раніше ESI – N 1, N 2 – RegN 21, EBX – базовий регістр, EAX – робочий регістр. Для другого блоку EDI – N 1, RegN 22 – N 2, , ECX – базовий регістр, EDX – робочий регістр. В даному випадку доцільно використати допоміжну макрокоманду для виконання однієї ітерації підстановки. Для восьмибітної підстановки така макрокоманда може мати наступний вигляд:

```
Iteration   Macro       @n
;@n - номер ітерації
            Mov         BL, AL                          ; для першого блоку
            Mov         CL, DL                          ; для другого блоку
            Mov         AL, LongKey[EBX+(@n shl 8)]     ;підстановка першого блоку
            Mov         DL, LongKey[ECX+(@n shl 8)]     ;підстановка другого блоку
            Ror         EAX, 8                          ; для першого блоку
            Ror         EDX, 8                          ; для другого блоку
            Endm
```

Тоді макрокоманда основного кроку криптографічного перетворення для двох блоків даних буде мати вигляд:

```
Step_Substit2       Macro  @m, last
;@m - номер ключа
;ESI - N1, RegN21 - N2 - для першого блоку
;EDI - N1, RegN22 - N2 - для другого блоку
                Mov     EAX,Key[@m shl 2]
                Mov     EDX, EDI
                Add     EAX, ESI
                Add     EDX, Key[@m shl 2]
```

```
        @j2v    =       0
                Rept    3
                Iteration               @jv2
        @jv2    =       @jv2+1
                Endm
                Mov     BL, AL
                Mov     CL, DL
                Mov     AL, LongKey[EBX+(3 shl 8)]      ;підстановка першого блоку
                Mov     DL, LongKey[ECX+(3 shl 8)]      ;підстановка другого блоку
                Rol     EAX, 3                          ;корекція для першого
блоку
                Rol     EDX, 3                          ;корекція для другого блоку
                Xor     EAX, RegN21             ;
                Xor     EDX, RegN22
                If      not last
                Mov     RegN21, ESI
                Mov     RegN22, EDI             ;
                Endif
        ;
                Mov     ESI, EAX                ;
                Mov     EDI, EDX
                Endm
```

Необхідні зміни вводяться і в макрокоманди базових циклів шифрування та розшифрування. За малоймовірними виключеннями будь-які дві сусідні команди таких макрокоманд на процесорах Pentium можуть виконуватись паралельно.

Розглянуті методи оптимізації криптографічних програм лягли в основу розробки криптографічного комплексу FastGost, що дозволило досягнути високої продуктивності при шифруванні та дешифруванні даних. Головну частину програмного комплексу FastGost (процедури шифрування та дешифрування, обчислення імітовставки) реалізовано у вигляді DDL - бібліотеки для Win32-сумісних операційних систем (Windows 95, 98, ME, NT, 2000 тощо), що забезпечує використання швидкодіючих процедур криптографічних перетворень за ГОСТ 28147-89 у поширених мовах програмування. Поточна версія програмного комплексу криптографічного перетворення дозволила отримати наступні результати швидкості шифрування та дешифрування у режимі гамування:

Для восьмибітних підстановок при послідовній обробці блоків

- UMC 486DX-2, 80 MHz. - 800 кб/с
- Intel Pentium 133MHz - 1750 кб/с

Для восьмибітних підстановок при паралельній обробці сусідніх блоків

- UMC 486DX-2, 80 MHz. - 1130 кб/с
- Intel Pentium 133MHz - 3200 кб/с

Ці значення продуктивності програм були отримані при тестуванні криптографічних перетворень файлу об'ємом 6 Мбайт.

Використання шістнадцятибітних підстановок за інших рівних умов дозволяє підвищити продуктивність порівняно з наведеною приблизно в 1.8 рази

III Висновки

ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.

Розглянуті реалізації методів оптимізації криптографічних програм за швидкодією, як показують результати тестування, можуть забезпечити синхронізацію процесу шифрування і передачі даних швидкісними каналами зв'язку в комп'ютерних мережах. Зростання об'ємів програм та даних є значними в порівнянні із традиційними методами реалізації алгоритмів ГОСТ 28147-89 [4] і є незначними в порівнянні із зростанням об'ємів системних програм OS Windows.

Література: 1. Михальчук В. М., Ровдо А. А., Рыжиков С. В. Микропроцессоры 80x86, Pentium. Архитектура, функционирование, программирование, оптимизация кода. - М.: "БИТРИКС", 1994. - 398 с. 2. Бердышев Е. Технология ММХ. Новые возможности процессоров P5 и P6. М. "ДИАЛОГ МИФИ", 1998 - 234 с. 3. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М. "Мир", 1979 - 527 с. 4. Мухачев В. А. К вопросу о разработке коммерческих криптосредств, использующих алгоритм ГОСТ 28147. - Праці науково-практичної конференції з питань криптографічного захисту інформації "УкрКрипт - 97", Одеса, 1997.

УДК 681.3.067:681.3.016

ТЕСТИРОВАНИЕ ДВОИЧНЫХ ВЕРОЯТНОСТНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ МЕТОДОМ БИНОМИАЛЬНОГО ПРЕОБРАЗОВАНИЯ

Тарас Левченко
Научно-технический комплекс "Импульс", г. Киев

Аннотация: **Проведен обзор некоторых методов тестирования двоичных вероятностных последовательностей (ДВП). Отмечены недостатки методов. Предложен метод тестирования, состоящий в анализе функции распределения сумм выборок. Метод проверен на 4 генераторах ДВП. Отмечена чувствительность метода к наличию неслучайной составляющей в ДВП.**

Summary: **The review of some methods of binary probabilistic sequences (BPS) testing is conducted. The lacks of methods are marked. The method of testing based on the analysis of the sampling sums of cumulative distribution function is offered. The method is tested for 4 BPS generators . The test-sensitivity to availability of non-random component in BPS is marked.**

Ключевые слова: **Информационная безопасность, двоичные вероятностные последовательности.**

I Введение

Двоичные вероятностные последовательности (ДВП)

$$\{b_N b_{N-1}..b_1 b_0\}, \tag{1}$$

где

$$b_i = \begin{cases} 1, p_1 = 0.5, \\ 0, p_0 = 0.5, \end{cases} \tag{2}$$

– некоррелированный бит в позиции номер i с дискретной плотностью распределения p, находят широкое применение для кодирования передаваемой информации при защите информационных ресурсов в сетях передачи данных [1]. Для получения ДВП обычно используют математические [2–7] или физические [8] методы генерации последовательностей нулей и единиц. Критерием использования генератора является неповторяемость фрагментов определенной длины при достаточно большом N.

Нечеткость определения понятия неповторяемости ведет к существованию